In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 1, lines 14 to 16 as follows:

--U.S. Provisional Application No. ~~60/___,___ (TI-29497)~~ 60/224,913 entitled PULL TRANSFERS AND TRANSFER RECEIPT CONFIRMATION IN A DATAPIPE ROUTING BRIDGE now U.S. Patent Application Serial No. 09/905,379; and--

Rewrite the paragraph at page 1, lines 16 to 18 as follows:

--U.S. Provisional Application No. ~~60/___,___ (TI-29499)~~ 60/224,586 entitled MULTIPROCESSOR NETWORK NODE FAILURE DETECTION AND RECOVERY now U.S. Patent Application Serial No. 09/904,991.--

Rewrite the paragraph at page 4, lines 19 to 27 as follows:

--6. The datapipe ~~Bus~~ bus ready signal improves inter-processor traffic management by autonomously propagating a not ready condition against the flow of data, to manage congestion of some transfer link segments without involvement of any chip resources. This autonomous traffic management is better than the hands-on traffic management of previous methods, because it releases valuable chip resources from having to be involved in traffic management and instead allows them to fully concentrate of the application tasks at hand.--

Rewrite the paragraph at page 6, line 22 to page 7, line 18 as follows:

--This application uses the descriptive name datapipe routing bridge or simply datapipe to describe a packet based communications peripheral connecting multiple processors without glue logic or CPU intervention. Figure 1 illustrates the makeup of a datapipe. It is composed of three building blocks transmitter 101, bridge ~~102~~

<u>103</u> and receiver ~~103~~ <u>102</u>. The main function of the bridge component is to provide high levels of connectivity between multiple digital signal processors without paying the penalties usually associated with inter-processor connections. Dedicated routing logic within the datapipe autonomously navigates data packets of programmable size along the shortest distance from the source processor to one or more destination processors. Transmitter 101 may transmit data packets via bridge 103 to one or both of the right and left ports. Transmitter 101 responds to transmit events and transmit interrupts from an associated data processor (not shown) to supply data from internal I/O memory 105 to bridge 103. Bridge 103 is capable of retransmitting a data packet received at one of the right or left ports to the other port. Bridge 103 may also transfer a received data packet to receiver 102 in addition to or instead of retransmission at the other port. The actions of bridge 103 are determined by a header of the data packet. Upon receipt of a data packet, receiver stores the received data in internal I/O memory 105 and may generate a receive event to the associated data processor. In the preferred embodiment the associated data processor is a digital signal processor.--

Rewrite the paragraph at page 9, lines 3 to 19 as follows:

--Figure 3 illustrates the three components of the datapipe hardware at each terminal node and their connection to the datapipe network in an example data transfer. The transmit controller 301 drives the packets from internal I/O RAM 302 out ~~the pins~~ <u>lines</u> 303 to the links connecting the digital signal processors. The communications bridge 304 routes each packet into or around each digital signal processor node on the network. For each packet routed into a node from the network, the receive unit 305 pushes the packet into the local I/O RAM 306 of the destination digital signal processor. Both of the two external ports of the bridge feature two unidirectional channels, one for input and one for output. Both transmitter and receiver can send communications events to the interrupt selectors in the associated digital signal processor. The transmitter can also respond to interrupts from the interrupt selector. The receiver can also send an interrupt directly to the transmitter.--

Rewrite the paragraph at page 9, line 27 to page 19, line 11 as follows:

--Figure 4 illustrates the datapipe within a conventional digital signal processor integrated circuit. Internal I/O RAM input buffers 405, when almost full, send an event to the chip direct memory access (DMA) unit to move the data into the level-2 (L2) main memory 401, where it can be accessed directly by the central processing unit core 400. Note that this application contemplates that central processing unit core 400 is a digital signal processor, however this invention is equally applicable to a general purpose data processor. Internal I/O RAM 405 of the datapipe is split into two independent blocks for simultaneous direct memory access unit and datapipe access. The direct memory access port servicing internal I/O RAM ~~406~~ 405 and the datapipe looks exactly like the other direct memory access ports driving the remaining chip peripherals. Figure 4 further illustrates conventional features of a digital signal processor including L2 instruction RAM 402, L2 data RAM 403, parameter RAM (PAR RAM), power down circuit (PWR DWN), phase locked loop circuit (PLL), first and second timers (TIMER0, TIMER1), a host port interface (HPI), two multi-channel buffered serial ports (McBSP0 and McBSP1) and a 32-bit external memory interface (EMIF32).--

Rewrite the paragraph at page 14, line 23 to page, 15, line 2 as follows:

--The use of the BLOCK tx_opcode is similar to an indirect addressing mode using the same processor analogy. The data that the BLOCK tx_opcode transmits ~~is~~ has its address embedded inside the BLOCK tx_opcode, but the data itself is residing in a different area of memory. A BLOCK tx_opcode causes the transmitter to transfer a block of data from a different local I/O RAM location, whose address has been previously loaded into the transmitter address register with other tx_opcodes preceding the BLOCK tx_opcode.--

Rewrite the paragraph at page 17, lines 3 to 16 as follows:

--To summarize the block transfer, the five 32-bit tx_opcodes (INITX, INITX, INITX, BLOCK and MSG) fetched from the tx_script, caused the transmitter to source an 11-byte packet into the datapipe network, consisting of two header rx_opcodes, eight bytes of data content followed by a single tail rx_opcode. Upon arrival at each of the two destination nodes, the three rx_opcode bytes BCAST 901, CHAN 902, and EVENT 904 were stripped off, leaving only the double word wide data content 903 to be written into the destination I/O RAM 915 at the current location 916 within the designated receive channels. Note, that the packet body for this transfer was not embedded inside the tx_script as in the previous (MSG) example, but instead was sourced from a separate, dedicated data location within source I/O ~~917~~ 905 RAM of the transmitting node.--

Rewrite the paragraph at page 35, line 20 to page 36, line 2 as follows:

--Figure 22 illustrates datapipe events, interrupts and configuration bits. Configuration of the datapipe is accomplished through a 27-bit CFG_BUS, which includes six inputs of reset and enable functions routed to the receiver, bridge, and transmitter, respectively. These are labeled 2201, 2202, and 2203 in Figure 22. A total of twenty-one monitor signals are routed back into the CFG_BUS 2200 I/O. These twenty one signals are: (a) two inputs from the transmitter labeled TX_STATE; and (b) seventeen event signals including TX_CIRC events (4), TX_STREAM events (2), RX_CIRC events (8), RX_STREAM events (3) and ⊕ (c) two interrupt signals INT_UNEXP and INT_TOUT. The two interrupt signals INT_UNEXP and INT_TOUT are also monitored.--